Asignatura	Datos del alumno	Fecha
Estructura de Computadores	Apellidos:	
	Nombre:	

Laboratorio #1: Simulación y optimización de un programa en un procesador escalar segmentado

Preparación para el laboratorio

Para poder realizar la práctica de laboratorio deberá usarse la aplicación MARS (MIPS Assembler and Runtime Simulator), que básicamente es un IDE para MIPS. El archivo con el cual podrá ejecutar la actividad es un archivo Executable Jar File (.jar) que tiene como nombre: Mars4_5 y que se encuentra en la siguiente carpeta:

Descripción del laboratorio

En esta práctica vamos a trabajar la ejecución de código Assembler que deberá pedir o mostrar por consola ciertos datos que permita la ejecución de los scripts propuestos. Esta actividad puede realizar de manera individual o grupal (máximo 2 estudiantes). Los scripts que se deben desarrollar son los siguientes:

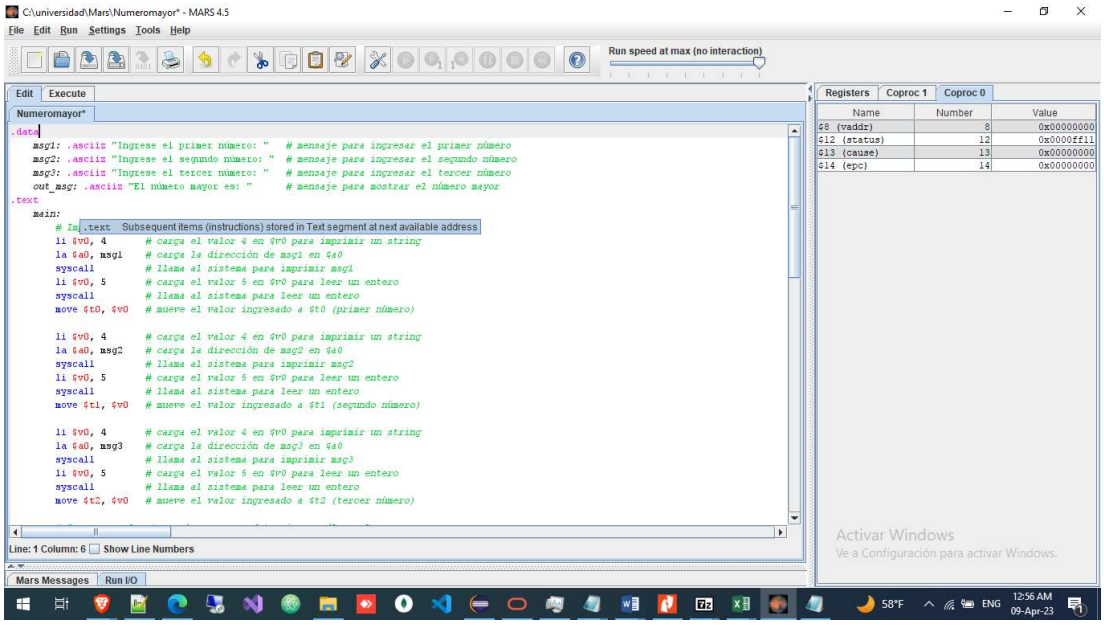
1. Número mayor (mínimo 3 números)
2. Número menor (mínimo 3 números)
3. Serie Fibonacci

- Para cada uno de los scripts propuestos para la actividad se debe realizar 3 capturas de pantalla:

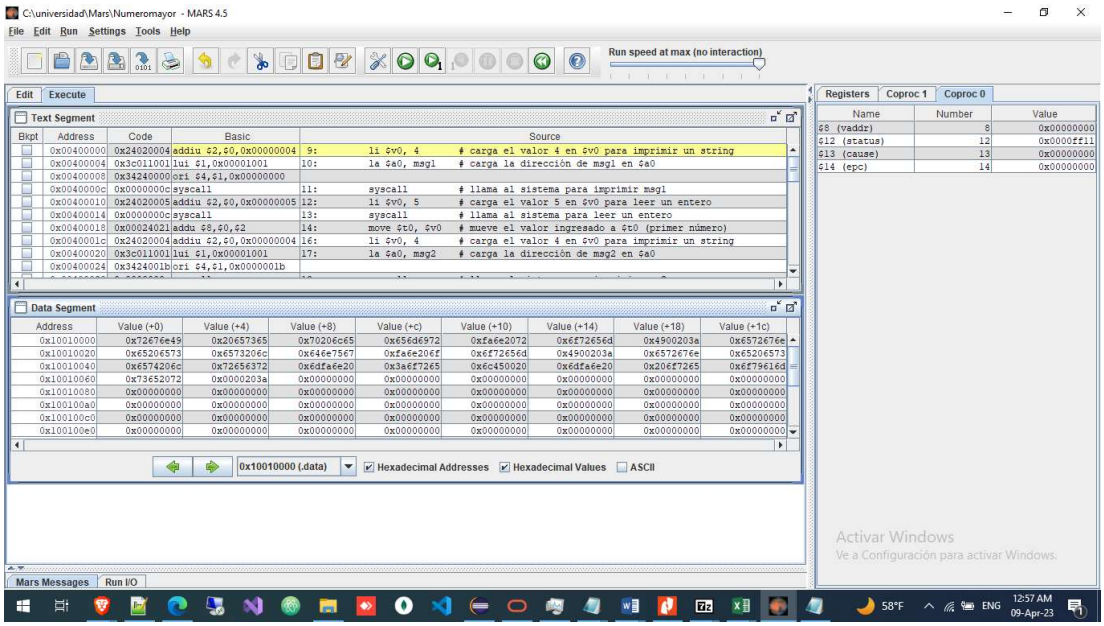
  PROGRAMA DE QUE DETERMINA EL NUMERO MAYOR DE TRES NUMEROS

  a. Antes de compilar

Asignatura	Datos del alumno	Fecha
Estructura de Computadores	Apellidos:	
	Nombre:	

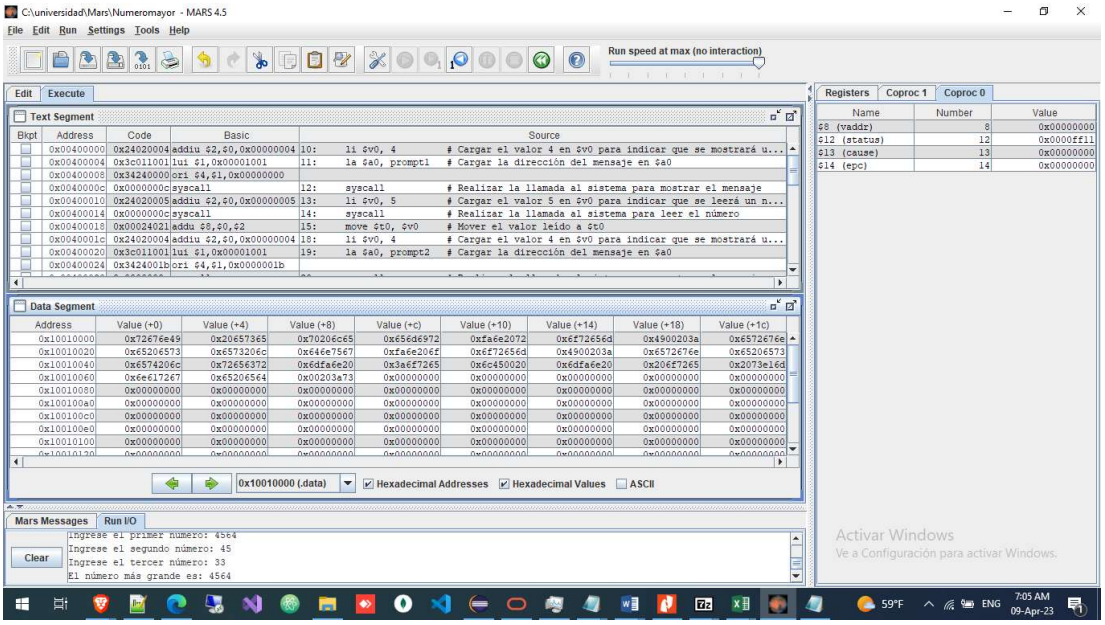


b. Después de compilar



c. Después de ejecutar

Asignatura	Datos del alumno	Fecha
Estructura de Computadores	Apellidos:	
	Nombre:	



b. El código en assembler (*.asm) de cada uno de los scripts.

```
.data
prompt1: .asciiz "Ingrese el primer número: "
prompt2: .asciiz "Ingrese el segundo número: "
prompt3: .asciiz "Ingrese el tercer número: "
output:  .asciiz "El número más grande es: "
# Definir la sección de código
.text
main:
    # Mostrar prompt1 y leer el primer número
    li $v0, 4       # Cargar el valor 4 en $v0 para indicar que se mostrará un mensaje en pantalla
    la $a0, prompt1   # Cargar la dirección del mensaje en $a0
    syscall         # Realizar la llamada al sistema para mostrar el mensaje
    li $v0, 5       # Cargar el valor 5 en $v0 para indicar que se leerá un número entero
    syscall         # Realizar la llamada al sistema para leer el número
    move $t0, $v0     # Mover el valor leído a $t0
```

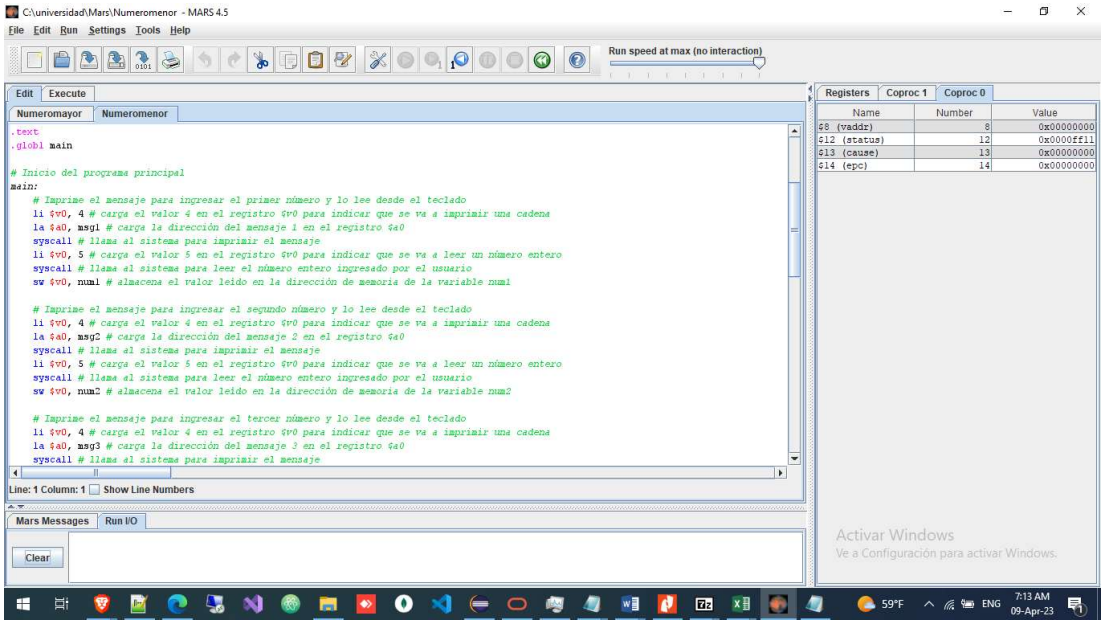
```
    # Mostrar prompt2 y leer el segundo número
    li $v0, 4       # Cargar el valor 4 en $v0 para indicar que se mostrará un mensaje en pantalla
    la $a0, prompt2   # Cargar la dirección del mensaje en $a0
    syscall         # Realizar la llamada al sistema para mostrar el mensaje
    li $v0, 5       # Cargar el valor 5 en $v0 para indicar que se leerá un número entero
    syscall         # Realizar la llamada al sistema para leer el número
    move $t1, $v0     # Mover el valor leído a $t1

    # Mostrar prompt3 y leer el tercer número
    li $v0, 4       # Cargar el valor 4 en $v0 para indicar que se mostrará un mensaje en pantalla
    la $a0, prompt3   # Cargar la dirección del mensaje en $a0
    syscall         # Realizar la llamada al sistema para mostrar el mensaje
    li $v0, 5       # Cargar el valor 5 en $v0 para indicar que se leerá un número entero
    syscall         # Realizar la llamada al sistema para leer el número
    move $t2, $v0     # Mover el valor leído a $t2

    # Comparar los números
    move $t3, $t0     # Asignar el primer número a $t3
    ble $t1, $t3, check2  # Saltar a "check2" si el segundo número no es mayor que el primero
    move $t3, $t1     # Asignar el segundo número a $t3 si es mayor
check2:
    ble $t2, $t3, print   # Saltar a "print" si el tercer número no es mayor que los dos primeros
    move $t3, $t2     # Asignar el tercer número a $t3 si es mayor
```

```
print:
    # Mostrar el número más grande
    li $v0, 4       # Cargar el valor 4 en $v0 para indicar que se mostrará un mensaje en pantalla
    la $a0, output    # Cargar la dirección del mensaje en $a0
    syscall         # Realizar la llamada al sistema para mostrar el mensaje
    move $a0, $t3     # Mover el número más grande a $a0
    li $v0, 1        # Cargar el valor 1 en $v0 para indicar que se mostrará un entero en pantalla
    syscall         # Realizar la llamada al sistema para mostrar el número
    li $v0, 10       # Cargar el valor 10 en $v0 para indicar que se finalizará el programa
    syscall         # Realizar la llamada al sistema para finalizar el programa
```

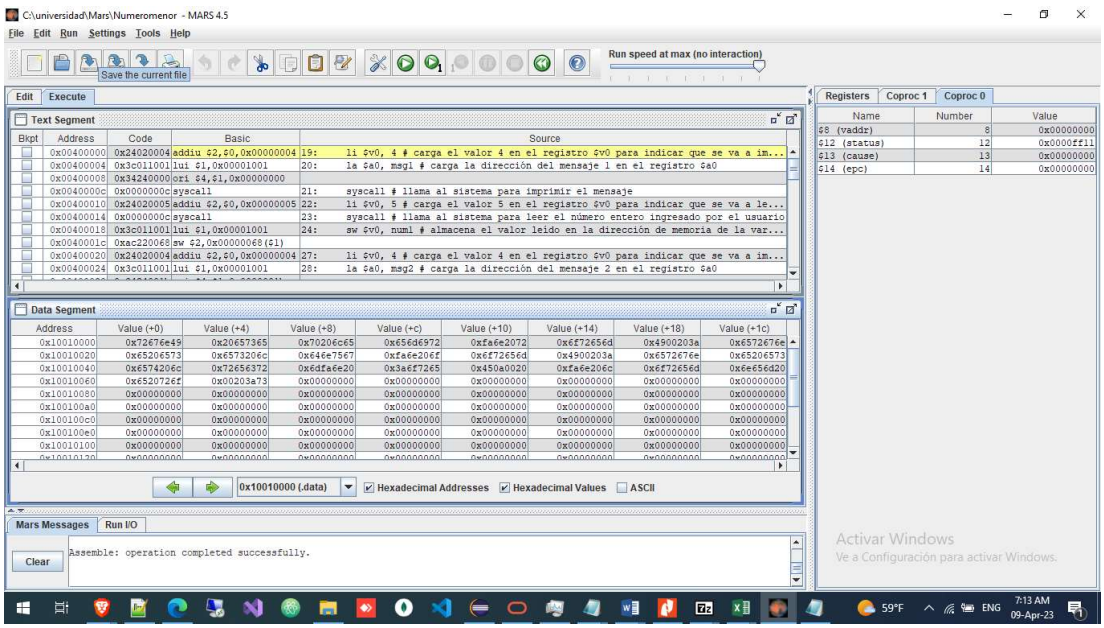
PROGRAMA DE QUE DETERMINA EL NUMERO MENOR DE TRES NUMEROS

a. Antes de compilar

Asignatura	Datos del alumno	Fecha
Estructura de Computadores	Apellidos:	
	Nombre:	

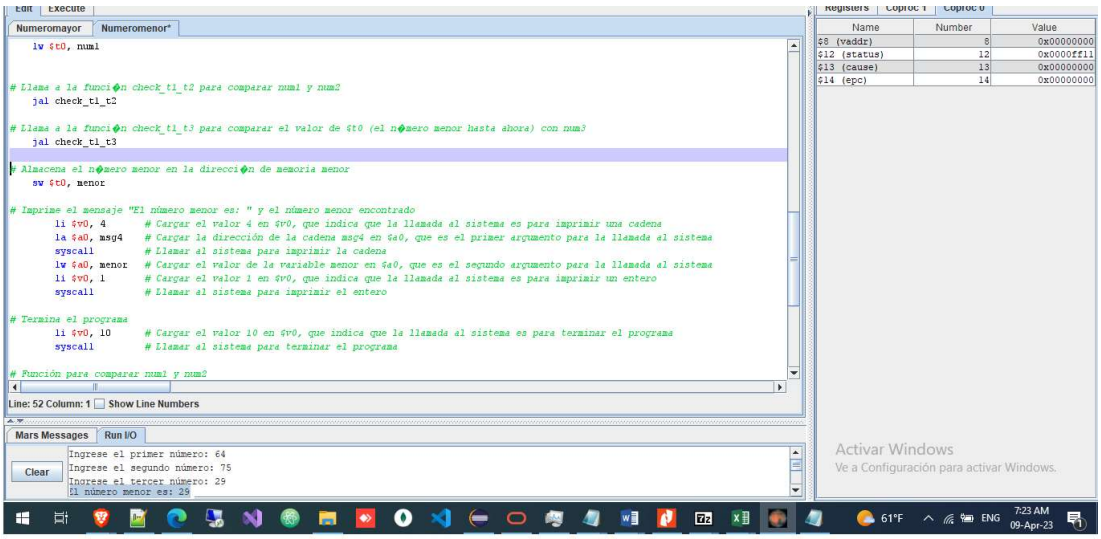


b. Después de compilar



c. Después de ejecutar

Asignatura	Datos del alumno	Fecha
Estructura de Computadores	Apellidos:	
	Nombre:	



b. El código en assembler (*.asm) de cada uno de los scripts.

```
# programa en MARS para encontrar el número menor entre tres número
.data
# Declaración de variables de datos
msg1: .asciiz "Ingrese el primer número: " # mensaje para el primer número
msg2: .asciiz "Ingrese el segundo número: " # mensaje para el segundo número
msg3: .asciiz "Ingrese el tercer número: " # mensaje para el tercer número
msg4: .asciiz "\nEl número menor es: " # mensaje para imprimir el número menor
num1: .word 0 # variable para almacenar el primer número
num2: .word 0 # variable para almacenar el segundo número
num3: .word 0 # variable para almacenar el tercer número
menor: .word 0 # variable para almacenar el número menor encontrado

.text
.globl main

# Inicio del programa principal
main:
    # Imprime el mensaje para ingresar el primer número y lo lee desde el teclado
```

```
li $v0, 4 # carga el valor 4 en el registro $v0 para indicar que se va a imprimir una cadena
la $a0, msg1 # carga la dirección del mensaje 1 en el registro $a0
syscall # llama al sistema para imprimir el mensaje
li $v0, 5 # carga el valor 5 en el registro $v0 para indicar que se va a leer un número entero
syscall # llama al sistema para leer el número entero ingresado por el usuario
sw $v0, num1 # almacena el valor leído en la dirección de memoria de la variable num1

# Imprime el mensaje para ingresar el segundo número y lo lee desde el teclado
li $v0, 4 # carga el valor 4 en el registro $v0 para indicar que se va a imprimir una cadena
la $a0, msg2 # carga la dirección del mensaje 2 en el registro $a0
syscall # llama al sistema para imprimir el mensaje
li $v0, 5 # carga el valor 5 en el registro $v0 para indicar que se va a leer un número entero
syscall # llama al sistema para leer el número entero ingresado por el usuario
sw $v0, num2 # almacena el valor leído en la dirección de memoria de la variable num2

# Imprime el mensaje para ingresar el tercer número y lo lee desde el teclado
li $v0, 4 # carga el valor 4 en el registro $v0 para indicar que se va a imprimir una cadena
la $a0, msg3 # carga la dirección del mensaje 3 en el registro $a0
syscall # llama al sistema para imprimir el mensaje
li $v0, 5 # carga el valor 5 en el registro $v0 para indicar que se va a leer un número entero
syscall # llama al sistema para leer el número entero ingresado por el usuario
```

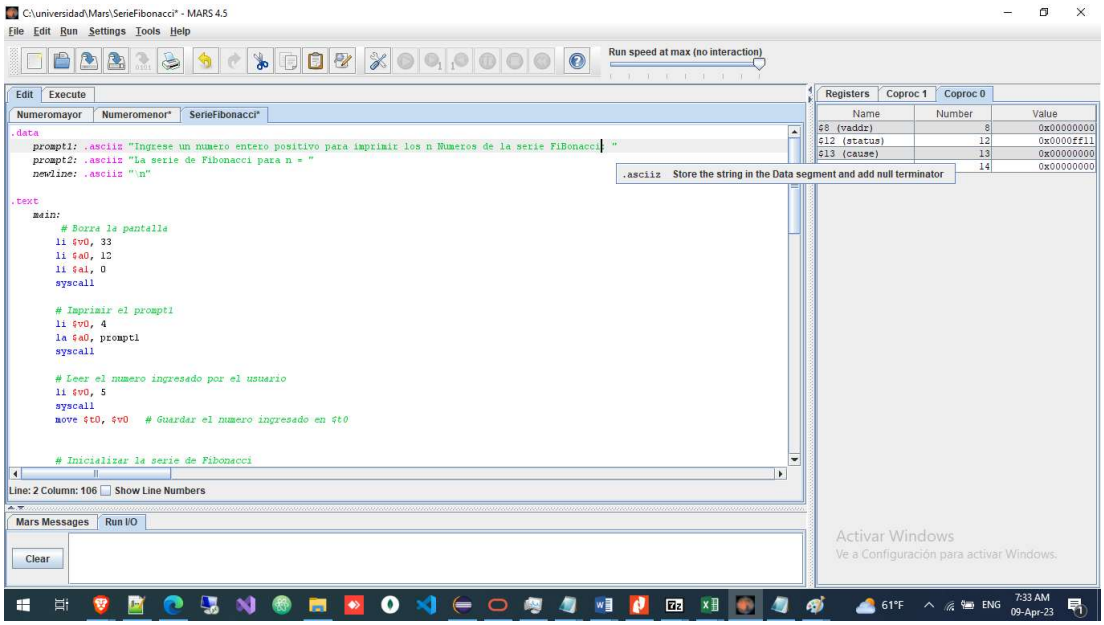
```
   sw $v0, num3 # almacena el valor leído en la dirección de memoria de la variable num3

   # Inicializa $t0 con el valor de num1
   lw $t0, num1


# Llama a la funcion check_t1_t2 para comparar num1 y num2
   jal check_t1_t2

# Llama a la funcion check_t1_t3 para comparar el valor de $t0 (el numero menor hasta ahora) con num3
   jal check_t1_t3

# Almacena el numero menor en la direccion de memoria menor
   sw $t0, menor

# Imprime el mensaje "El número menor es: " y el número menor encontrado
        li $v0, 4       # Cargar el valor 4 en $v0, que indica que la llamada al sistema es para imprimir una cadena
        la $a0, msg4    # Cargar la dirección de la cadena msg4 en $a0, que es el primer argumento para la llamada al sistema
        syscall         # Llamar al sistema para imprimir la cadena
        lw $a0, menor   # Cargar el valor de la variable menor en $a0, que es el segundo argumento para la llamada al sistema
        li $v0, 1       # Cargar el valor 1 en $v0, que indica que la llamada al sistema es para imprimir un entero
        syscall         # Llamar al sistema para imprimir el entero
```

```
# Termina el programa
        li $v0, 10      # Cargar el valor 10 en $v0, que indica que la llamada al sistema es para terminar el programa
        syscall         # Llamar al sistema para terminar el programa

# Función para comparar num1 y num2
# Si num1 es mayor que num2, $t0 = num2, de lo contrario $t0 = num1
        check_t1_t2:
        lw $t1, num2    # Cargar el valor de num2 en $t1
        ble $t0, $t1, set_t0_t1 # Comparar $t0 con $t1. Si $t0 es menor o igual, saltar a set_t0_t1, de lo contrario continuar.
        move $t0, $t1   # Si $t1 es menor, cargar $t1 en $t0
        set_t0_t1:
        jr $ra          # Devolver el control a la llamada anterior

# Función para comparar el número menor actual ($t0) con num3
# Si num3 es menor que $t0, $t0 = num3
        check_t1_t3:
        lw $t1, num3    # Cargar el valor de num3 en $t1
        ble $t0, $t1, set_t0_t1 # Comparar $t0 con $t1. Si $t0 es menor o igual, saltar a set_t0_t1, de lo contrario continuar.
        move $t0, $t1   # Si $t1 es menor, cargar $t1 en $t0
        jr $ra          # Devolver el control a la llamada anterior
```

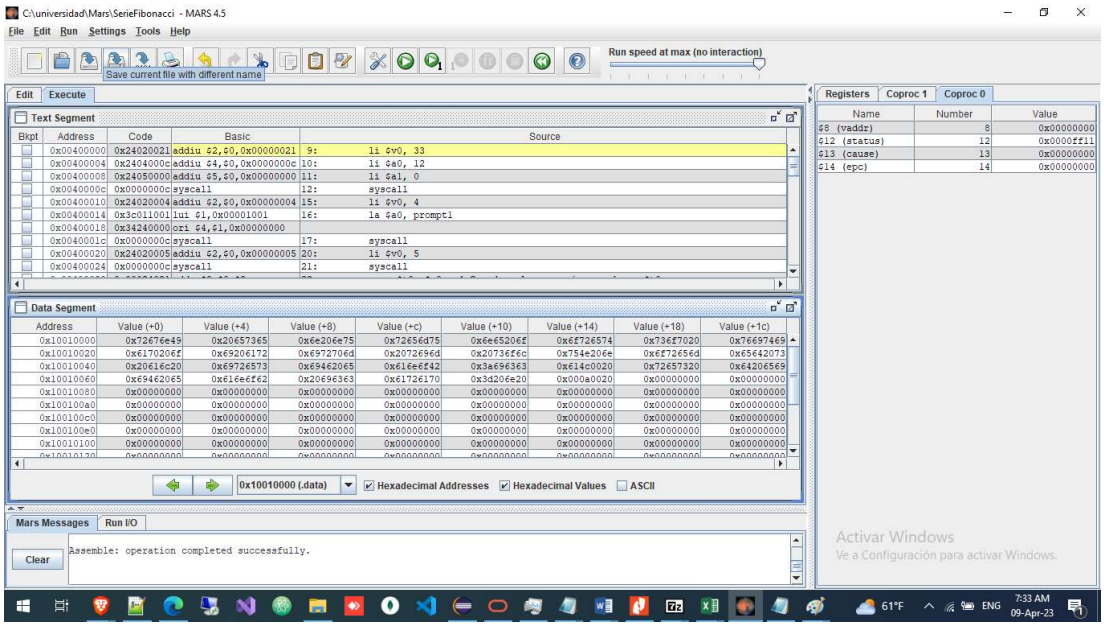
Asignatura	Datos del alumno	Fecha
Estructura de Computadores	Apellidos:	
	Nombre:	

1. PROGRAMA QUE DETERMINA LOS NUMERO DE LA SERIE FIBONACCI

a. Antes de compilar

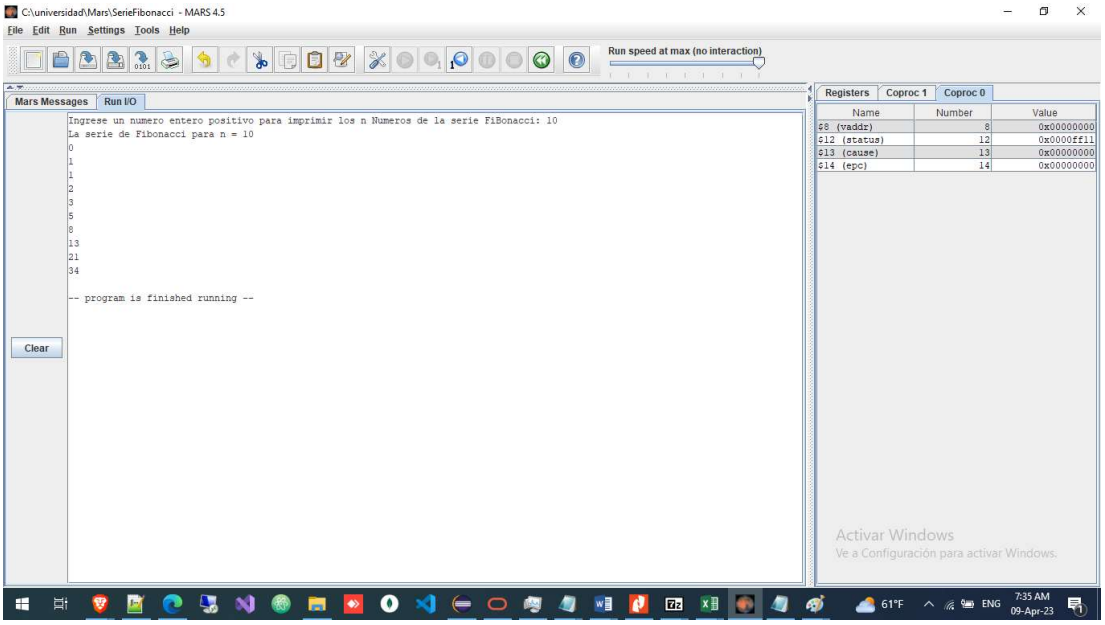


b. Después de compilar



c. Después de ejecutar

Asignatura	Datos del alumno	Fecha
Estructura de Computadores	Apellidos:	
	Nombre:	




```
.data
    prompt1: .asciiz "Ingrese un numero entero positivo para imprimir los n Numeros de la serie FiBonacci: "
    prompt2: .asciiz "La serie de Fibonacci para n = "
    newline: .asciiz "\n"

.text
    main:
                                        # Borra la pantalla
                                        li $v0, 33
                                        li $a0, 12
                                        li $a1, 0
                                        syscall

        # Imprimir el prompt1
        li $v0, 4
        la $a0, prompt1
        syscall
```

```
# Leer el numero ingresado por el usuario
li $v0, 5
syscall
move $t0, $v0   # Guardar el numero ingresado en $t0


# Inicializar la serie de Fibonacci
li $t1, 0       # F(0) = 0
li $t2, 1       # F(1) = 1
li $t3, 2       # i = 2

# Imprimir el prompt2 con el valor de n
li $v0, 4
la $a0, prompt2
syscall


move $a0, $t0   # Pasar el valor de n como argumento para imprimir
li $v0, 1
syscall

li $v0, 4
la $a0, newline      # Imprimir un salto de l?nea
syscall

                                    li $v0, 1
                                    li $a0, 0
                                    syscall
```

```
    li $v0, 4
    la $a0, newline      # Imprimir un salto de linea
    syscall

                                li $v0, 1
                                li $a0, 1
                                syscall

    li $v0, 4
    la $a0, newline      # Imprimir un salto de linea
    syscall


# Imprimir la serie de Fibonacci
loop:
    blt $t3, $t0, body   # Saltar a la etiqueta body si i < n
    j exit               # Si i >= n, salir del loop
body:
    add $t4, $t1, $t2    # Calcular F(i) = F(i-1) + F(i-2)
    move $t1, $t2        # Actualizar F(i-2) = F(i-1)
    move $t2, $t4        # Actualizar F(i-1) = F(i)
    li $v0, 1
    move $a0, $t4        # Imprimir F(i)
    syscall
    li $v0, 4
    la $a0, newline      # Imprimir un salto de linea
    syscall
    addi $t3, $t3, 1     # Incrementar i en 1
```

```
    j loop          # Volver al inicio del loop
# Salir del programa
exit:
    li $v0, 10
    syscall
```

https://github.com/GalaxsoMundo/Laboratorio1